Evolutionary City Generator

About

By Ho-Wan To

Prepared for Morphogenetic Programming Module: Assignment A6.

Presentation Date: 27th April 2018

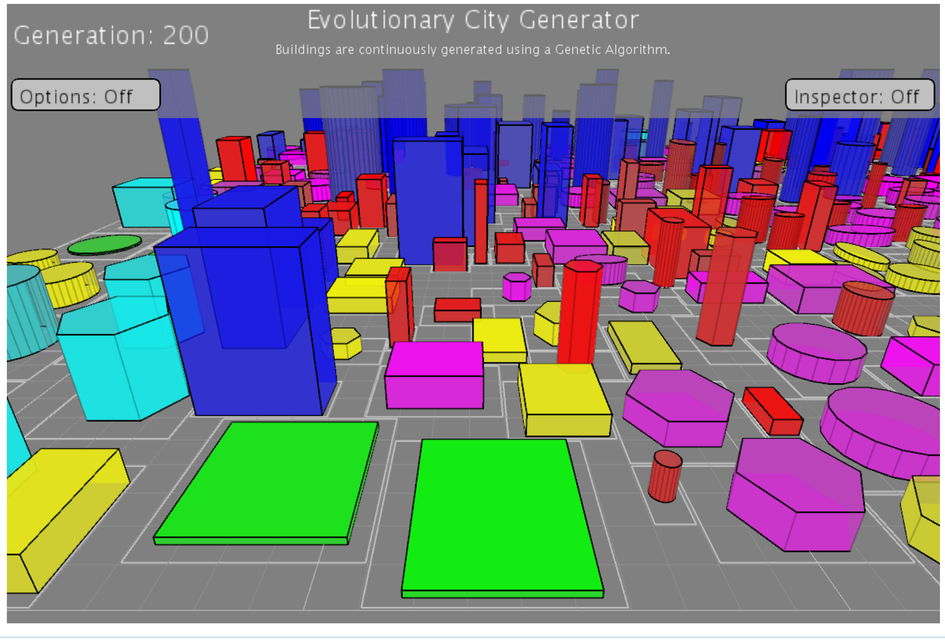
Introduction

This processing sketch creates a city by generating buildings based on a evolutionary approach, using a **Genetic Algorithm**. Several buildings are created at each generation, each inheriting properties from parent buildings. The objective is to form clusters of similar buildings whilst maintaining variety. Buildings can die off based on their fitness, so buildings that have a good fitness are more likely to survive and pass on their properties to new buildings, as analogous to the process of natural selection. The user can adjust the fitness function scale and bias using scrollbars.

Common methods for procedural city generation typically consists of creating the buildings in one or several pre-defined steps. One example is CityEngine, which firstly generates a road layout, then defines blocks of buildings, followed by a one-off generation of all buildings (likely using Perlin noise with randomisation) . In real cities, neighbourhoods generally grow from specific sectors often which have a specific purpose, forming natural clusters of neighbourhoods. Taking London as an example, there is a cluster of retail at Oxford Street, and tall Office towers at Canary Wharf. In addition, cities continuously evolve, with new neighbourhoods appearing and replacing out dated sectors.

My approach aims to more closely resemble this natural process of growing clusters of buildings with similar characteristics and usages. It allows any arrangement of existing buildings to be adopted, and evolved into natural clusters by killing off buildings that are do not fit in well within the surrounding area.

Image



Pseudocode

1. Create grid of cells.
2. Create initial seed buildings (5 no. as defined by user).
3. Generate 3 new buildings:
   a. Select random footprint size (Gene 0) at a location adjacent to an existing building, checking that position does not clash with any existing buildings.
   b. Select building usage (Gene 1) at random from allowable usages for given footprint size.
   c. For new building, find all buildings within influence area (100m); if > 3 buildings nearby, then select parents from inside influence area; otherwise any 2 parents.
   d. New building inherits Genes 2 to Gene 5 from either parent via crossover of genes, with a chance of a random mutation (5%).
   e. Phenotype generated based on gene values sets the appearance and physical properties of the building.
4. Evaluate fitness function for each building and scale.
   e. Fitness calculated by comparing to all nearby buildings within influence area and measuring distance (stored in a HashMap). A positive score is given for same building usage, and negative score for different building usage. Scores are divided by the distance and summed.
   f. Fitness is scaled using an “Exponential” and “Bias” value, which can be adjusted by user using scrollbars in the Options window.
5. Check if building dies; Poor fitness increases chance of death.
6. Update building count and list containing all adjacent cells.
7. Repeat steps 3 to 6 at each generation.

Sketch info

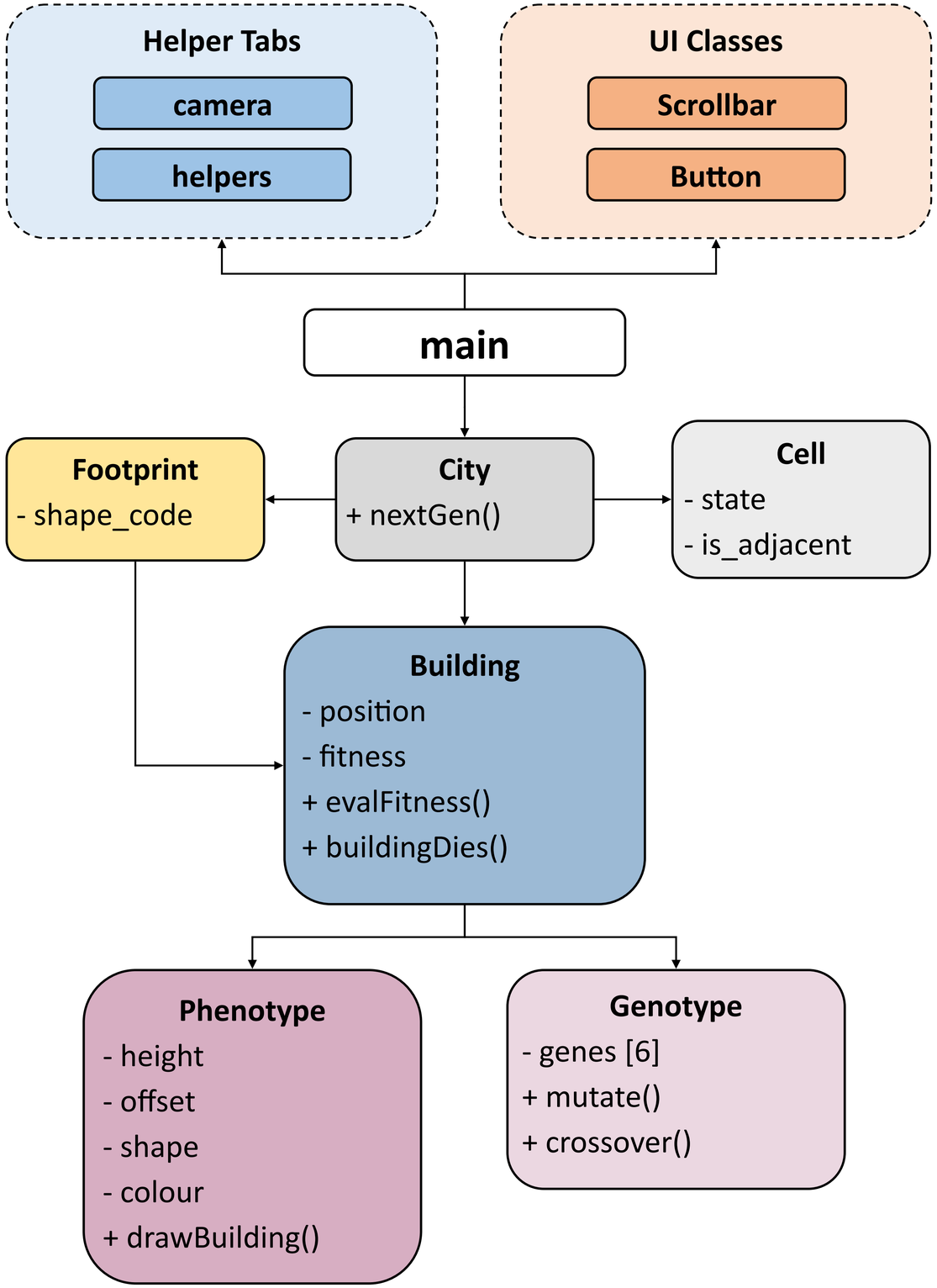
Total lines of code: ~ 1800 , 64 commits on GitHub

Total number of functions = 102

All code is written by myself, with only the following snippets taken from elsewhere, adapted as follows:

1. drawCylinder() function in phenotype.pde. Function adapted to extrude ellipse instead of truncate cylinder. Source: *http://vormplus.be/blog/article/drawing-a-cylinder-with-processing.*
2. Scrollbar class in scrollbar.pde. Adapted for user to set min and max values associated with slider, and display value above slider.

Class Diagram



Genes

Gene 0

Shape Code: 0 to 11

0 = (1 x 2), 1 = (2 x 1)

2 = (2 x 2), 3 = (2 x 3), etc.

Gene 1

Usage: 0 to 5

0 = Residential

1 = Retail

2 = Office

3 = Industrial

4 = Leisure

Gene 2

Height: 0 to 8

Each usage has it's own height range, e.g.

Resi: min = 5, max = 45

Office: min = 20, max = 100

,etc.

Gene 3

Offset: 0 to 8

0 to 3: reduce footprint size

4 to 7: offsets to one side

Gene 4

Shape: 0 to 8

0, 5, 6, 7 = Cuboid

1 = Hexagon

2 = Cylinder

3 = Ellipse (extruded)

4 = Split level box

Gene 5

Colour Modifier: 0 to 8

Modifies RGB colour value by (gene value * 12)

Fitness Function

Raw fitness

For all buildings within in-fluence area:

If (same type):

f += 2 * (100 / dist)

Else (different type):

f -= 1 * (100 / dist)

Fitness Bias

f_s = (a * f) + b

default: a = 0.02, b = 0

min: a = 0.01, b = - 0.5

max: a = 0.01, b = + 0.5

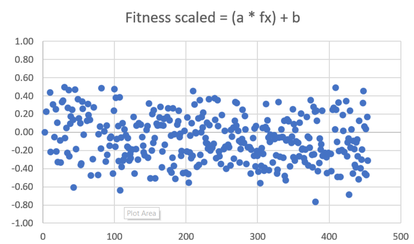
Fitness Exponential

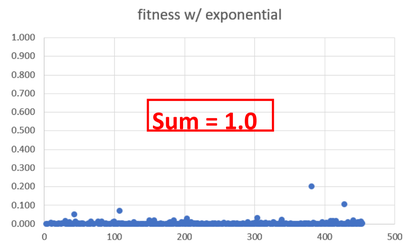
f_se = (f_s) ^ c

default: c = 6

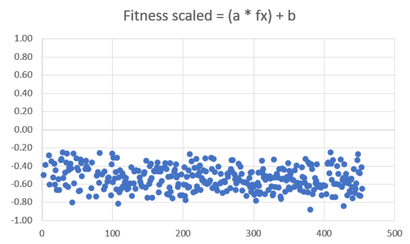
min c = 1, max c = 11

Fitness function values for all buildings at Generation 150 (339 Buildings)

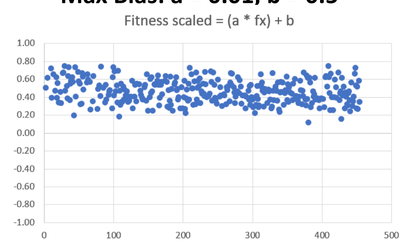




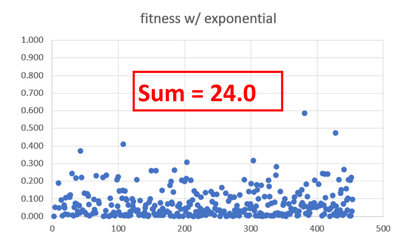
Min Bias: a = 0.01, b = - 0.5



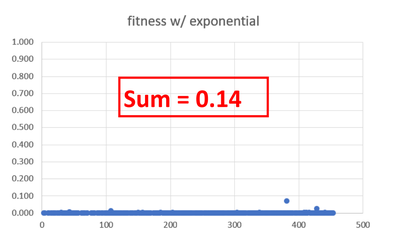
Max Bias: a = 0.01, b = 0.5



Exponential: c = 2

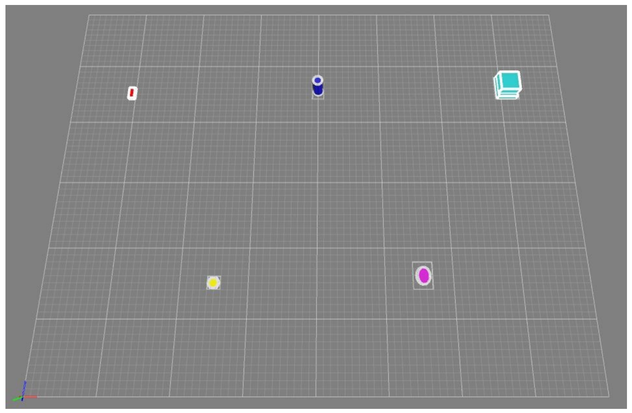


Exponential: c = 10

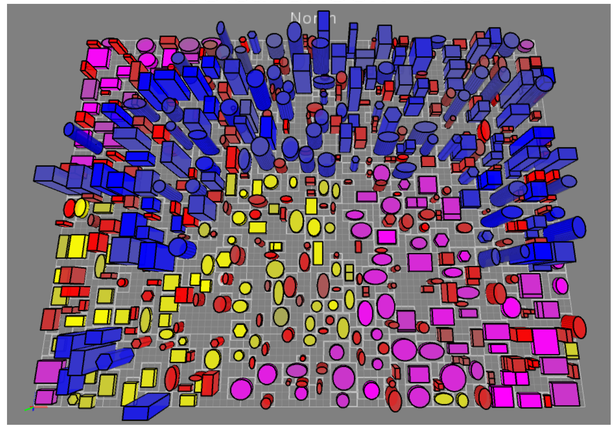


Base Case - Figures

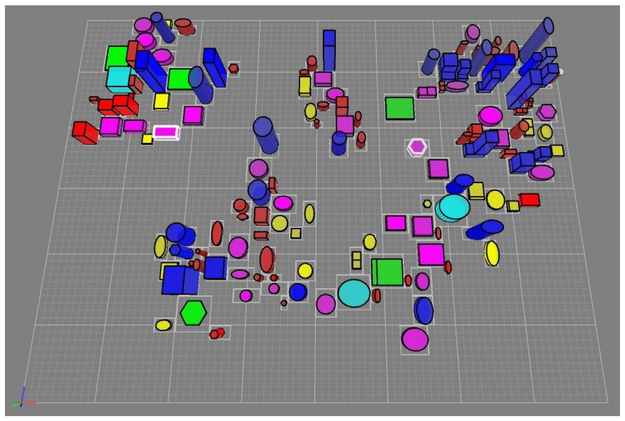
Base Case: Gen 0



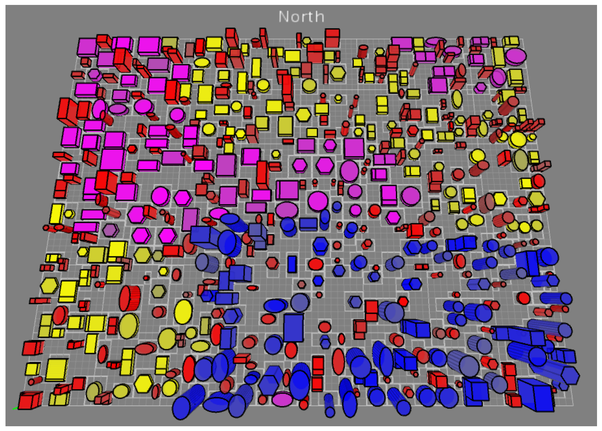
Base Case: Gen 1000



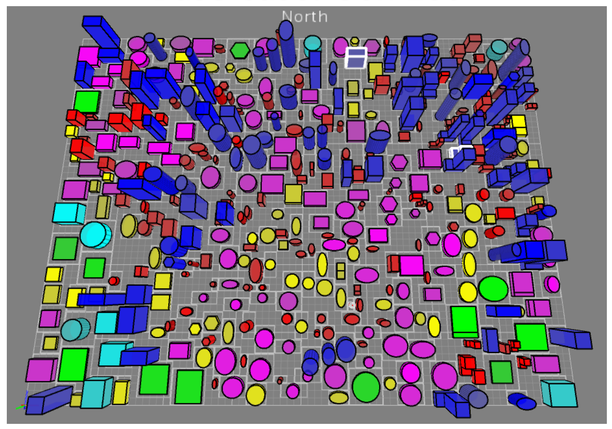
Base Case: Gen 50



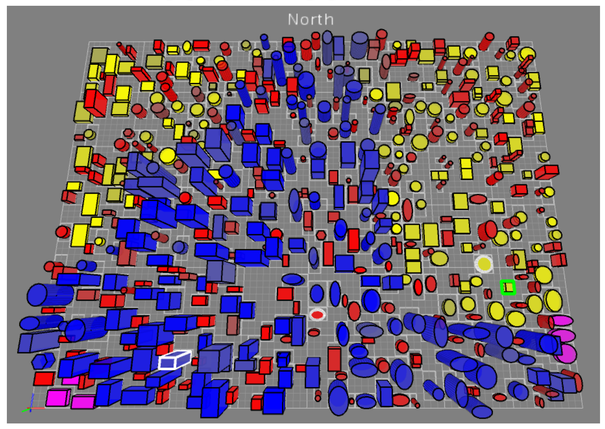
Seed 1: Gen 1000



Base Case: Gen 200

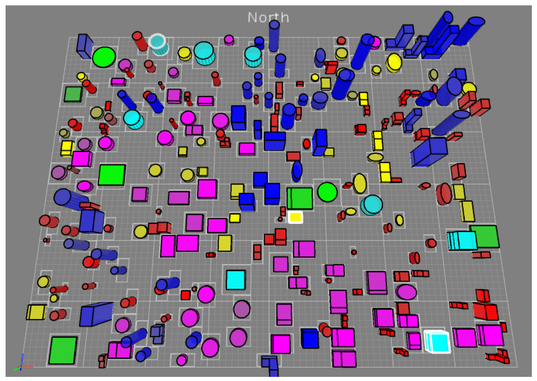


Seed 2: Gen 1000

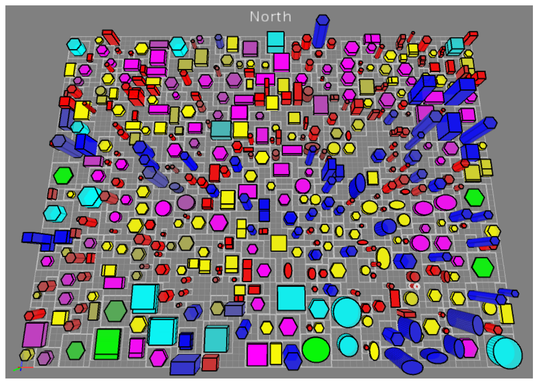


Variations

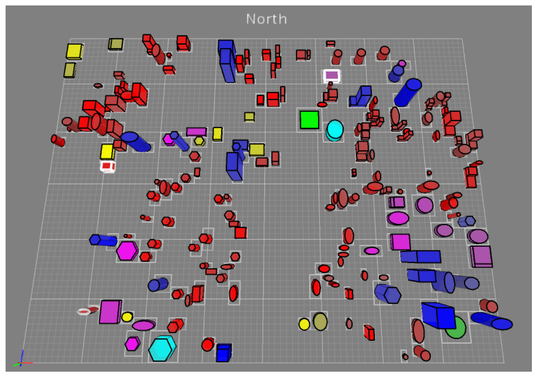
Exponential = 3: Gen 200



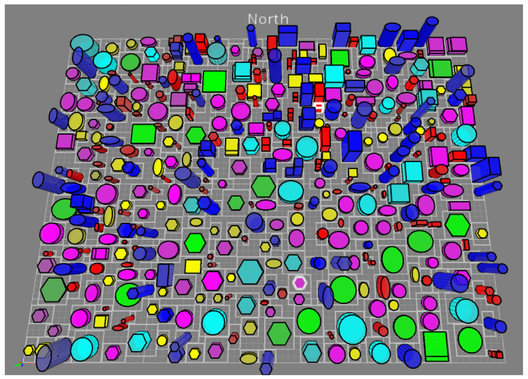
Exponential = 10: Gen 200



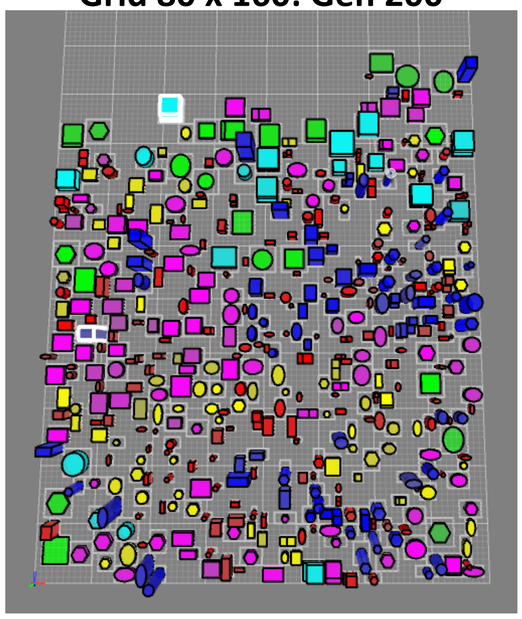
Bias = -1: Gen 200



Bias = +1: Gen 200



Grid 80 x 160: Gen 200



Grid 80 x 160: Gen 1000